SAM-FS -- LSC's New Solaris-Based Storage Management Product

Kent Angell
LSC, Inc.
380 South 200 West
PO Box 657
Farmington UT 84025
kent@lsci.com
801-451-9704

I have been involved with the computer industry for more than 20 years, and I've identified a few constants. These are change, growth, and an exponentially increasing volume of data to be managed.

My industry niche is in data management, and for that reason I am very focused on the growth of data volume, storage density, data granularity, and I/O bandwidth. I think that the experience of Cray users over the years is a relevant case study.

In the early 1980s, Seymour Cray completed work on the Cray 2, a compact supercomputer that used a revolutionary liquid cooling system. The Cray 2 could create new data at the rate of (about) 5 to 10 megabytes / second. The processing speed was great enough that data generation and use outpaced the machine’s I/O channels, and the system had to be idled every few hours to allow for export of newly created data, and import of a new data set. The cycle was repeated several times a day. Data transfer was a bottleneck, and the best that could be hoped for was 1 to 2 gigabytes / day.

The Y-MP class of Cray machines provided a greatly increased I/O bandwidth, and at the same time the capacity of both vector and scalar machines was increased so that data could be created at 10, 20, 30 ... megabytes per second. All of the previously compute-bound problems became I/O bound, and this created a logistics problem, which is -- how do we move and store all of this data?

Automated Storage Management

Well, the traditional approach has been to keep our current stuff on magnetic disk, and store less current information on tape. (Stuff, that's a technical term we use to describe the other guy's digital data.) This scheme worked OK, as long as we had "manageable" data sets, and enough operators to keep the data moving. But when data volumes get into the range of 1 to 10 gigabytes per day, we must have automation to handle the sheer volume of data, to provide transparent access, to give us around-the-clock operation, and to manage access control, that is -- security.

IBM marketed one of the first automated storage management systems, based on their proprietary DFHSM software. And, the emergence of UNIX and open systems computing led to a second generation of storage management applications, such as Bump, developed at NRL, and UniTree, developed at Lawrence Livermore.

These were followed by a wave of architecturally similar products, such as Cray's Data Migration Facility (DMF), which falls into the Bump class, and products in the UniTree mold that use proprietary file systems, and operate in parallel with the UNIX file system. Both methodologies offered customers improved functionality, and satisfied many of their customer's requirements at the time.

But the marketplace has changed. UniTree and other like products were traded like baseball cards, and were not valued or supported by the new owners, as strongly as by the old.

There is much that is new in today's market, and much remains the same. Basic customer requirements have not changed; they need stability, scalability, performance, and support for the newest storage subsystems. Customers want to know that their data is secure and accessible, with redundant copies for disaster recovery. But the check-off list of required functionality has grown significantly, today's customers want more functionality than any one software product can provide, and they want it low-cost, with guaranteed updates and tech. support -- forever.

A major issue for all customers is vendor stability, which means: will the software provider be there to support the product next year - and in five years. Many place their hopes in large companies that are presumed to be more viable, but this is inconsistent with the dynamics of the computer industry, where growth, change and innovation are fueled by products developed at start-ups and small companies. In fact, both large and small companies in the computer industry are volatile and offer volatile product lines. Whoever the provider, customers are advised to secure software that works for their environment and is supported by established systems integrators.

SAM-FS from LSC

Which brings me to my company, LSC, which stands for Large Storage Configurations. Our HSM product, called SAM-FS, is the happy result of an eclectic marriage of the best elements of the old paradigm with a new generation of code, functionality and performance. SAM-FS is a robust, high-performance storage and archive manager, operating under Sun's Solaris 2.X operating system.

I'll have more to say about performance later, but it's important to make a point about it here. Performance is important in all aspects of HSM operation -- not just in I/O transfer rates.

LSC has designed performance into every part of SAM-FS; into file system restoration for example. In an actual test performed by DLR in Germany (that's Germany's NASA) a SAM-FS file system and a competitor's system were restored from backup media after a simulated interruption. The competitor's product required about 1-1/2 days per

100,000 files restored; SAM-FS took less than a minute per 100,000 files. We didn't do this test, the customer (DLR) designed the test and carried it out.

Don't you think that performance like that will be important to <u>your</u> users or customers if they are trying to get back into production after an interruption, maybe a server disk crash?

SAM-FS is a full featured HSM that operates as a file system on Solaris-based machines. The SAM-FS file system provides the user with all of the standard UNIX system utilities and calls, and it adds some new commands, i.e. archive, release, stage, sls, sfind, and a family of maintenance commands. The system also offers enhancements to the standard UFS, i.e. high performance virtual disk read and write, control of disk through an extent array, and the ability to dynamically allocate block size. This allows for very fast disk access, up to 2X faster than ufs. SAM-FS supports all RAID levels and the use of 3rd party file systems such as Veritas and On-line Disk Suite (ODS).

SAM-FS provides "archive sets", which are groupings of data to be copied to secondary storage. Archive sets can be defined (controlled) as to number of copies, when and where each copy will be stored, how long each copy will be retained, file size included (max. and min. sizes), and by VSN for each copy in the set.

As with other HSM systems, SAM-FS migrates files onto secondary media. In practice, as soon as a file is written to disk, SAM-FS will make copies onto secondary media. These files then become candidates for release from disk cache. The archiving process can be automatic or explicitly driven. This may not sound all that revolutionary, but there are some very neat things going on:

First, one to four copies of a file are dynamically and automatically written to secondary media, either automatically or by specific command. And SAM-FS provides parallel threaded operation so that all files can be written to I/O devices simultaneously.

Second, data is written to secondary media with the metadata included, and can be read independently using standard *tar* on any UNIX system.

OK, now we have the file on disk and secondary storage; what now? The SAM-FS *releaser* utility can be tasked specifically to release a file, or
group of files, with immediate operation. Or the *releaser* can be programmed to release files according to predetermined criteria. Files can be specified for immediate release after archiving, or can be tagged release-never, which means they are backed-up (archived) but never leave the disk. Now, the released files are off of cache and reside only on secondary storage in a tape library, jukebox, or on media stored on shelves, possibly in a vault somewhere.

Files are <u>staged</u> onto cache to get them back. Think of the system as a virtual disk; when the user accesses a file, he wants it as soon as possible. To do this fast you must tune the system to take advantage of the media performance, which means to do parallel

I/O at full speed. Stage requests are organized by media type and VSN, with the queue organized for most efficient access to the media. New requests are added to the queue dynamically and are placed in the most logical place based on media type and VSN. Once a request is satisfied, and if no other requests are pending, the tape or MO remains in the drive for a user-defined period of time. Then it is rewound and put away.

File access can be specified *stage-never* so as to bypass the disk cache, allowing large file access directly from secondary media, without disturbing the file mix in cache (3rd party transfers).

Users can access all or any part of a file, specifying the start of data (byte offset) and the number of bytes to retrieve. Only the specified data will be returned.

Files can be archived leaving a stub on disk cache. This allows the file to be opened and read without staging it onto disk.

When files are modified they get a new date and time and are archived as a new file. The pointer to the old version is deleted, and the media now has a hole.

Now, having holes in your media isn't all bad, and I suggest that you keep them. Because as long as that hole is present in the media, the older version of the file that the hole represents is still accessible. Sooner or later, though, you may want to recycle media. The SAM-FS *recycler* will copy the remaining files onto new media, and you can reuse the old media, or you can keep it until you don't need access to the older file versions anymore. SAM-FS provides utilities and procedures to access the older file versions.

Scalability

SAM-FS is a richly scalable storage management system. It can manage N file systems on one server, where N is a very large number limited by 64 bit architecture. The system can manage millions of files per system, though this is limited today by the speed of UNIX and its utilities. Later this year, LSC will implement a new search algorithm that removes logical and performance restrictions on the number of files.

Currently, SAM-FS supports tape and MO libraries from all major vendors, including Grau robots with mixed media tapes and StorageTek libraries with Timberline and Redwood drives.

Performance and Testing

LSC has tested the more popular tape devices under load to validate vendor claims and to determine actual performance with SAM-FS:

	Native	Compressed
DLT 2000	1.2 MByte/sec.	2.0 MByte/sec.
DLT 4000	1.5 MByte/sec.	3.5 MByte/sec.
DLT 7000	not yet available for test	
3490E	3.5 MByte/sec.	5.0 MByte/sec.
Redwood (FW SCSI)	9.5 MByte/sec.	14.0 MByte/sec.

SAM-FS scales in performance from one drive to simultaneous use of multiple drives as follows:

1 DLT 2000	1.2 MByte/sec.	1 Redwood	9.0 MByte/sec.
2 DLT 2000	2.4 MByte/sec.	2 Redwood	18.0 MByte/sec.
3 DLT 2000	3.6 MByte/sec.	3 Redwood	27.0 MByte/sec.
.	.	.	.
.	.	.	.
N DLT 2000	n(1.2) MByte/sec.	n Redwood	n(9.0) MByte/sec.

This test assumes the use of multiple SCSI channels. If all drives are on the same SCSI bus, then performance will suffer.

SAM-FS was also tested to determine added overhead:

Disk writes	+ 1%
Disk reads	+ 2%
Tape writes	+ 0.1%
Tape reads	+ 0.1%

For more information about SAM-FS testing and performance contact one or both of the following:

DLR Deutsche Forschungsanstalt fur Luft ung Raumfahrt e.V.
German Remote Sensing Data Center (DFD)
82234 Oberpfaffenhofen, Germany
Phone: 49 8153 282 623
E-mail: willi@dfd.dlr.de
rattei@dfd.dlr.de
how@dfd.dlr.de
Contact Names: Wilhelm Wildegger, Willi Rattei and John How

JIC-PAC ISO
P.O. Box 500
Pearl Harbor, HI 96860
Phone: (808) 471-7272
E-mail: djp@pixi.com
Contact: Dale Podoll

Summary

It is a fact of life in the storage management software business that everybody wants something more than we can deliver today. They love what they see but they also want something different or more of it. This is more of a job description than a problem, LSC is a customer-driven software development company. We add the new requested features and enhancements to our release schedule if we think its a good idea, and we get a better product over time.

In the last release several customer-requested features were added:

API: Interface to access SAM-FS from a user application. Both client and server versions are provided.

Grow fs: Enables additional disk cache devices to be added to a file system as it grows. Additional disk devices can be added without system reinitialization.

Stage All: Provides Associative Staging. Files in a common directory with this control set are all staged when any one of the set is accessed.

In the next release are more customer-requested features:

Checksum Verification: Enables users to verify that data on removable media has not been altered.

Recycling for Archive Sets: Allows recycling based on archive set thresholds, in addition to recycling based on robot thresholds.

API Enhancements: Making the API jump through hoops.

The Last Word

A major reason that customers want HSM is for disaster recovery, and SAM-FS has some unique capabilities for data recovery in case of catastrophic failure:

First, if the disk cache dies, the system can be reinitialized and back on line in a matter of 1 to x minutes depending on the number of files and speed of the tape device reading the last inode dump. Average performance is 1 minute / 100K files in the SAM-FS file system.

Second, if a file is damaged, the archive copy is used. If the archive copy is damaged, SAM-FS will look for a second copy. Older versions (holes) can be accessed using the SAM-FS interface.

Third, if the primary server installation is destroyed, the system can be reinitialized using replacement hardware and a backup copy of the archive maintained specifically for this contingency in a remote vault.

Last, it is important to tune the storage management system with disaster recovery in mind, i.e. files that have not been migrated to secondary storage will be lost.

People who try SAM-FS like it, so I'd like to ask you to give it a try. We have demo software available if you want to wring it out.